PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 9/00 | A1

(11) International Publication Number: WO 97/26731

(43) International Publication Date: 24 July 1997 (24.07.97)

(21) International Application Number: PCT/US97/00640

(22) International Filing Date: 16 January 1997 (16.01.97)

(30) Priority Data:
08/585,765 16 January 1996 (16.01.96) US

(71) Applicant: RAPTOR SYSTEMS, INC. [US/US]; 69 Hickory Drive, Waltham, MA 02154 (US).

(72) Inventors: LEVESQUE, Roger, H.; 71 Windsor Drive, Tewksbury, MA 01876 (US). KIRBY, Alan, J.; 17 Mendelssohn Drive, Hollis, NH 03049 (US).

(74) Agent: FEIGENBAUM, David, L.; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).

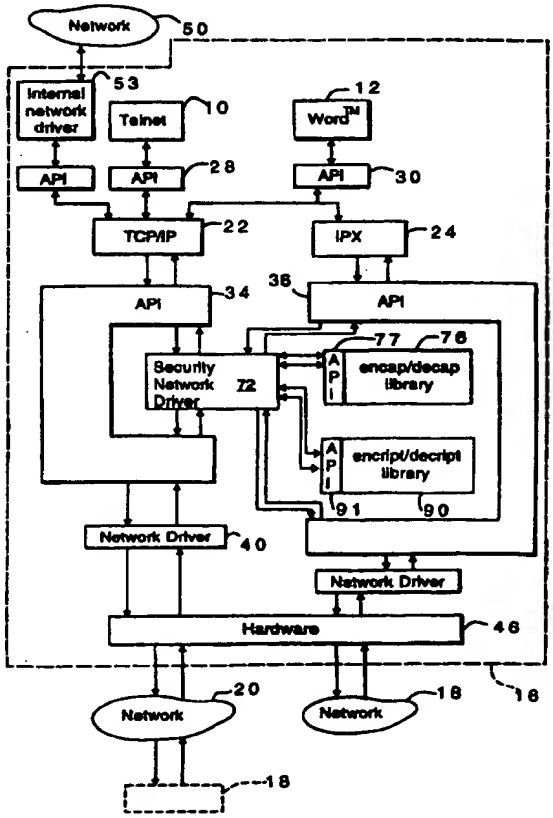
(81) Designated States: AU, CA, IL, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: DATA ENCRYPTION/DECRYPTION FOR NETWORK COMMUNICATION

(57) Abstract

Security network driver software (72) is inserted between the network protocol TCP/IP and the corresponding network driver (40). Security measures are performed upon the network packets before the network packets are passed to the network protocol TCP/IP (22) from the application programming interface (34). A security network driver (72) is used along with two encryption/decryption libraries (76, 90) for added data security. A network driver (40) and additional hardware (46) are also used in communication with the networks (20, 21, 50).



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

DATA ENCRYPTION/DECRYPTION FOR NETWORK COMMUNICATION

Background

This invention relates to data encryption/decryption for network communication.

Referring to Fig. 1, while executing a variety of software applications, 10, 12, 14, for example, Telnet 10 or Microsoft™, Inc. Word™ 12, computers 16 and 18 may exchange data over networks 20, 21, for example, a telephone company network, a private network, or a public network such as the internet or X.25. The applications communicate using network protocols 22, 24, 26, for example, transmission control protocol/internet protocol (TCP/IP) 22 or internet packet exchange (IPX) 24, through application programming interfaces 28, 30, 32. Through application programming interfaces 34, 36, 38, the network protocols communicate with network drivers 40, 42, 44 to direct network interface hardware 46, 48 to transfer data over the networks.

While on a network, data being transmitted, including the addresses of the source and destination computers 16, 18, is accessible to others who may be monitoring the network. For security, the data is often encrypted before being sent on the network.

Referring also to Fig. 2, for additional security, firewall computers 16, 18, which have direct access to a network 20 may be used to prevent unauthorized access to internal/private networks 50, 52. For example, when an internal network driver 53 within firewall computer 16 receives data from an internal computer 54 that is destined for a computer 56 on a public network, it encrypts the data and the addresses of source computer 54 and destination computer 56. Computer 16 then prepends

to the encrypted data a new IP header including its own address as well as the address of a destination computer, which may also be a firewall computer, e.g., computer 18.

When a firewall computer receives a network packet from the network, it determines whether the transmission is authorized. If so, the computer examines the header within the packet to determine what encryption algorithm was used to encrypt the packet. Using this algorithm and a secret key, the computer decrypts the data and addresses of the source and destination computers 54, 56 and sends the data to the destination computer. If both the source and destination computers are firewall computers, the only addresses visible (i.e., unencrypted) on the network are those of the firewall computers. The addresses of computers on the internal networks, and, hence, the internal network topology, are hidden. This has been termed "virtual private networking" (VPN).

Encrypting/decrypting data has been performed by complex security software within applications or, to simplify the applications, encrypting/decrypting has been performed within the protocol stack of network protocols.

Summary

In general, in one aspect, the invention features a method for processing network packets communicated on a network. Network packets are passed between a network protocol and a network driver via an application programming interface, and security measures are performed on the network packets before the network packets are passed to the network protocol from the application programming interface.

Implementations of the invention may include one or more of the following features. The security measures may be performed on the network packets before the network packets are passed to the network driver from the

application programming interface. The security measures may be performed by a security network driver. The application programming interface may pass network packets to the security network driver and, after performing the security measures, the security network driver may pass the network packets back to the application programming interface. The network may be a public network. The security measures may include encapsulating the network packets before the network packets are transferred to the network, decapsulating the network packets after the network packets are received from the network, encrypting the network packets before the network packets are transferred to the network, and decrypting the network packets after the network packets are received from the network. The security measures may be selectable, and may be selectable through libraries. The libraries may include an encapsulation/decapsulation library and an encryption/decryption library. The method may also include, before passing network packets between the network protocol and the network driver via the application programming interface, passing the network packets between an application and the network protocol or between an internal network driver and the network protocol.

In general, in another aspect, the invention features a method for processing network packets communicated on a network. The network packets are passed from a network protocol to an application programming interface, and then from the application programming interface to a security network driver. Security measures are performed on the network packets which are then passed back from the security network driver to the application programming interface. The network packets are then sent to a network driver which sends the network packets over the network.

Implementations of the invention may include one or more of the following features. Before passing network packets from the network protocol to the application programming interface, the network packets are passed from an application or an internal network driver to the network protocol. The method may also include receiving network packets over the network, passing the network packets from the network driver to the application programming interface, passing the network packets from the application programming interface to the security network driver, performing security measures on the network packets, passing the network packets from the security network driver back to the application programming interface, and passing the network packets from the application programming interface to the network protocol. The method may further include passing the network packets from the network protocol to an application or an internal network driver.

In general, in another aspect, the invention features a method for use with a network protocol application programming interface. Network packets are passed to a security network driver from the application programming interface and security measures are performed on the network packets. The secure network packets are then passed back to the application programming interface.

Implementations of the invention may include one or more of the following features. Network packets may be passed between a network protocol and the application programming interface, and network packets may be passed between a network driver and the application programming interface. Before passing network packets to the security network driver, the method may include altering a road map to allow network packets to be passed between

the application programming interface and the security network driver. The network may be a public network.

In general, in another aspect, the invention features a network packet processor including a network protocol and an application programming interface coupled with the network protocol and configured to pass network packets with the network protocol. The network packet processor also includes a security network driver coupled with the application programming interface which is configured to pass the network packets with the application programming interface and perform security measures on the network packets. A network driver coupled with the application programming interface is configured to pass the network packets with the application programming interface.

Implementations of the invention may include one or more of the following features. The network packet processor may also include an application coupled with the network protocol and configured to pass the network packets with the network protocol, and an internal network driver coupled with the network protocol and configured to pass the network packets with the network protocol.

Advantages of the invention may include one or more of the following. Providing a separate security network driver for encrypting/decrypting and encapsulating/decapsulating data and addresses simplifies the applications and network protocols. Newly developed security features may be implemented by modifying only the security network driver, instead of the complex security software in each application or the protocol stacks of each network protocol. In addition, because the security network driver provides the user with those security features required by the user, users are not limited to those applications and protocols that

implement the necessary security features, and any application or protocol may be used without modification. Further, the security network driver may access any available encryption/decryption library and encapsulation/decapsulation library. Because the applications and protocols do not access these libraries, the user's choice of applications and protocols is not limited by the available libraries.

Other advantages and features will become apparent from the following description and from the claims.

Description

Fig. 1 is a block diagram of two computers connected together through two networks.

Fig. 2 is a block diagram of two firewall computers and networks.

Fig. 3 is a block diagram of a computer including a security network driver.

Fig. 4 is a flow chart of encapsulation and encryption.

Figs. 5 and 6 are block diagrams of network packets.

Fig. 7 is a flow chart of decryption and decapsulation.

Fig. 8 is a block diagram of virtual tunnels.

Fig. 9 is a block diagram of a computer network.

Fig. 10 is a flow chart of tunnel record generation.

Fig. 11 is a flow chart of tunnel record updating.

As seen in Fig. 3, security network driver software 72 is inserted between network protocol TCP/IP 22 and corresponding network driver 40. The security network driver encrypts information before it is sent on the network by the network driver and decrypts information received from the network by the network driver before the information is sent to the network

protocol. As a result, after choosing a security network driver with the required security features, users may freely choose among available applications and network protocols regardless of the required level of security and regardless of the available encryption/decryption libraries and without having to compromise their security needs. Moreover, the chosen applications and network protocols need not be modified. To change the level of security, the user may simply chose another security network driver or modify the current security network driver.

Generally, a computer's operating system software defines a "road map" indicating which applications may communicate with each other. To insert a security network driver between a network protocol and a network driver, the road map is altered. The vendor of the operating system software may make the road map available or the road map may be determined through observation and testing. Once the road map is altered, functions such as send and receive, between the network protocol and the network driver are diverted to the network security driver to encrypt data before it is sent on the network and to decrypt data when it is received from the network.

Referring to Figs. 3 and 4, as an example, to send data from computer 16 to computer 18 on the internet, Telnet 10 issues (step 60) a send call to TCP/IP 22 through network protocol API 28. The send call includes a network packet 62 (Fig. 5) having a header 64 and data 66. The header includes information such as the addresses of the source and destination computers and the type of application that sent the data. The network protocol then issues (step 68) a send call to the network driver API which, in accordance with the altered road map, issues (step 70) a send call to a security network driver (SND) 72.

The security network driver issues (step 74) an encapsulate call to an encapsulate/decapsulate library 76 through an API 77. In one example, the encapsulate/decapsulate library uses the swIPe IP Security Protocol created by J. Ioannidis of Columbia University and M. Blaze of AT&T™, Inc. which is described in an Internet Draft dated December 3, 1993 and incorporated by reference. Referring also to Fig. 6, the encapsulate call generates a new network packet 78 in accordance with the swIPe protocol. The new packet includes a header 80, a swIPe protocol header 82, and data 84. According to options within the swIPe protocol, header 80 may be the original header 64 (Fig. 5), in which case, data 84 is the original data 66, or header 80 may be a new header including the address of a source firewall computer, e.g., computer 16 (Fig. 2), and a destination computer which may also be a firewall computer, e.g., 18. Where header 80 is a new header, data 84 includes the entire original network packet 62 (Fig. 5).

After encapsulating the network packet, the security network driver issues (step 88, Fig. 4) an encryption call to an encryption/decryption library 90 (Fig. 3) through an API 91. Library 90 encrypts a portion 92 of the encapsulated network packet including data 84 and part of swIPe protocol header 82. Header 80 (Fig. 6) is not encrypted. Thus, if, according to options within the swIPe protocol, header 80 is the original header 64 (Fig. 5), then the addresses of the source and destination computers are visible on the internet. On the other hand, if header 80 is a new header including the addresses of firewall computers, then the addresses of internal source and destination computers are encrypted and not visible on the internet.

Library 90 may be of the type sold by RSA Data Security™, Inc. of Redwood City, California and may encrypt the data according to an RSA algorithm such as RC2 or RC4 or according to a federal information processing standard (FIPS) such as data encryption standard (DES).

The security network driver then issues (step 94) a send call, including the encapsulated/encrypted network packet, to the API, and the API, in accordance with the altered road map, issues (step 96) a send call to a network driver, e.g., network driver 40. The network driver then causes hardware 46 to transmit (step 98) the encapsulated/encrypted network packet on the network.

Referring to Figs. 3 and 7, the network drivers of each computer 16, 18 (Figs. 2 and 3) maintain a database of addresses to which they will respond. For example, when network driver 40 receives (step 100) a properly addressed network packet from network 20, the network driver issues (step 102) a receive call to corresponding network protocol API 34. In accordance with the altered road map, the API issues (step 104) a receive call to security network driver (SND) 72 which issues (step 106) an authorization call to encapsulate/decapsulate library 76 through API 77. Library 76 examines the unencrypted portion of swIPe header 82 (Fig. 6) to determine (step 108) whether it is proper. If it is not proper, an error (step 110) is flagged.

If the header 82 is not a swIPe header, then the security network driver issues a receive call to the API including the unaltered packet.

If the swIPe header is proper, the security network driver issues (step 112) a decryption call to encryption/decryption library 90 through API 91. A portion of the unencrypted swIPe protocol header includes a policy identification (id) field 113. The policy id

field indicates the encryption algorithm used to encrypt the data. Library 90 uses a secret key that was previously exchanged between the computers and the encryption algorithm to decrypt data 84.

After decryption, the security network driver issues (step 114, Fig. 7) a digital signature check call to encapsulate/decapsulate library 76. The swIPe protocol header includes a digital signature 86. The digital signature is a unique number calculated using the data in the network packet, the secret key, and a digital signature algorithm. Library 76 recalculates the digital signature and compares (step 116) it to digital signature 86 in the network packet. If the network packet is tampered with during transmission and any data within the packet is changed, then the digital signature in the packet will not match the digital signature generated by the receiving computer and an error (step 118) will be flagged.

If the signatures match, then the security network driver issues (step 120) a receive call to the API which issues (step 121) a receive call to the TCP/IP network protocol including only the original network packet 62 (Fig. 5, data 66 and addresses of the source and destination computers 64). If (step 122) the network packet is destined for computer 16, then TCP/IP issues (step 124) a receive call to an application 10, 12 and if the network packet is destined for a computer on an internal network, e.g., computer 54 (Fig. 2) on network 50, then TCP/IP issues (step 126) a receive call to internal network driver 53 which then sends (step 128) the data to the internal computer.

Referring to Fig. 8, the policy id field may be used to create virtual tunnels 140, 142 between firewall computers 146, 148 on internet 152. When computer 146 receives a network packet, it checks the policy id to

determine which "tunnel" the packet came through. The tunnel indicates the type of encryption algorithm used to encrypt the packet.

Multiple tunnels 140, 142 may connect two computers 146, 148 and each tunnel may use a different encryption algorithm. For example, tunnel 140 may use the RC2 encryption algorithm from RSA Data Security™, Inc. while tunnel 142 uses the FIPS DES encryption algorithm. Because the RC2 encryption algorithm is less secure and requires less computer processing time than the FIPS DES standard, users may send a larger number of network packets requiring less security over tunnel 140 as opposed to tunnel 142. Similarly, predetermined groups of users or computers may be restricted to sending their packets over particular tunnels (effectively attaching a packet filter to each tunnel).

The tunnel may also indicate where the packet is to be sent. Primary firewall computers 16, 18 store information about the internal path of each tunnel in a tunnel database. When computer 146 receives a packet whose policy id indicates that the packet came through a tunnel that ends at computer 146, e.g., tunnel 142, computer 146 decapsulates and decrypts the packet and sends the decrypted packet over internal network 154 to the proper destination computer in accordance with the decrypted destination address. When computer 146 receives a packet whose policy id indicates that it came through a tunnel that does not end with computer 146, e.g., tunnel 140, computer 146 does not decapsulate and decrypt the packet. Instead, computer 146 sends the encrypted packet to internal firewall computer 158 in accordance with the tunnel database.

Internal firewall computer 158 also has a tunnel database in which the internal path of any tunnels connected to computer 158 are stored. As a result, when

computer 158 receives a packet whose policy id indicates that it came through a tunnel that ends with computer 158, e.g., tunnel 140, it decapsulates and decrypts the packet according to the policy id and sends the decrypted packet over internal network 160 to computer 162 in accordance with the decrypted destination address.

The only addresses visible on the internet and on internal network 154 are the addresses of the firewall computers 146, 148, and 158. The address of internal computer 162 and, hence, the network topology of network 160 are protected on both the internet and internal network 154.

The tunnel databases provide the firewall computers 146, 148, and 158 with information as to the internal path of the tunnels. Thus, if computer 162 was another firewall computer, computer 146 may modify the destination address of packets received on tunnel 140 to be the address of computer 162 to cause computer 158 to send the packet directly to computer 162 without checking the policy id field.

Encapsulating/decapsulating and encrypting/decrypting network packets may require a large portion of a computer's processing power. Creating virtual tunnels using the policy id field allows the encapsulating/decapsulating and encrypting/decrypting of network packets to be spread across several computers. For example, computer 146 may decapsulate and decrypt network packets destined for computers connected to internal network 154 while computer 158 may decapsulate and decrypt network packets destined for computers connected to internal networks 154 and 160. Similarly, computer 146 may encapsulate and encrypt network packets sent from computers connected to internal network 154 while computer 158 may encapsulate and encrypt network

packets sent from computers connected to internal networks 154 and 160.

The Kerberos Key Distribution Center components of Kerberos Network Authentication System created under project Athena at Massachusetts Institute of Technology, defines one method of providing computers with secret keys. Referring to Fig. 9, computer 130 is termed the "trusted" computer, and before computers 132 and 134 may transfer encrypted data to each other over network 136, both computers send a request to trusted computer 130 for a secret key. For a more detailed description of the Kerberos Key Distribution Center, see RFC1510 (request for comment) "Kerberos Network Authentication Service" by J. Kohl & B. Neuman, September 10, 1993, which is incorporated by reference.

Referring back to Fig. 2, to transfer secure (i.e., encapsulated and/or encrypted) network packets between two computers, operators of the two computers may verbally exchange a secret key for each tunnel between the computers and then manually initialize the computers to transfer data by generating a tunnel record including a secret key for each tunnel between the two computers. Firewall computers are typically managed by skilled technicians capable of generating tunnel records. Typical users have non-firewall computers and may wish to transfer encapsulated/encrypted data with a firewall computer. To avoid requiring that a typical user generate tunnel records and instead of having a separate trusted computer provide secret keys to two computers, a firewall computer 16, 18 may provide secret keys to other computers.

Referring also to Fig. 10, when a user wishes to transfer packets between his/her computer and a firewall computer, the user requests (step 170) a password (a one-time pad) from the firewall operator. The operator then

generates (step 172) tunnel records for each tunnel over which the user's computer and the firewall computer may transfer network packets. The operator also stores (step 174) the password given to the user on the firewall computer. The user installs (step 176) the security network driver (SND) software on his/her computer and runs (step 178) a configuration program. The configuration program prompts (step 180) the user for the password and sends (step 182) a configuration request to the firewall computer.

The firewall computer identifies (step 184) the user's computer as the sender of the request and notifies the user's computer of the available tunnels by sending (step 186) the complete tunnel records, including secret keys, associated with each tunnel to the user's computer. The tunnel records are sent through network packets that are encrypted using the password and the encryption algorithm. Afterwards, the firewall deletes (step 188) the password, and further network packets are transmitted between the two computers through the available tunnels and encrypted according to the secret key associated with each tunnel.

Referring to Fig. 11, generally, each time the user's computer accesses (step 190) the internet, a new internet address is assigned. The firewall computer needs to know the new address in order to update the tunnel records. To notify the firewall computer of the new internet address, each time the user's computer accesses the internet, the configuration software issues (step 192) a connect request to the firewall computer. The firewall computer identifies (step 194) the computer and may prompt the user for a user name and a user password. If the user name and password are authorized (step 196), the firewall updates (step 198) the tunnel records with the internet address sent as part of the

connect request. The configuration software also updates (step 200) the non-firewall computer's tunnel records with the computer's new internet address.

Other embodiments are within the scope of the following claims.

For example, instead of encapsulating the network packets using the swIPe protocol header, other internet security algorithms may be used.

Although the security network driver was described with respect to send and receive functions, APIs from different manufacturers, for example, Sun™, Inc. and Microsoft™, Inc., include a variety functions, and the security network driver is designed to respond to each possible function.

The security network driver may also be simultaneously connected to multiple network protocols, e.g., both TCP/IP 22 and IPX 24, as shown in Fig. 3.

What is claimed is:

1. A method for processing network packets communicated on a network, comprising:

passing network packets between a network protocol and a network driver via an application programming interface; and

performing security measures on the network packets before the network packets are passed to the network protocol from the application programming interface.

2. The method of claim 1, further comprising:

performing security measures on the network packets before the network packets are passed to the network driver from the application programming interface.

3. The method of claim 1, wherein the security measures are performed by a security network driver.

4. The method of claim 3, wherein the application programming interface passes network packets to the security network driver and, after performing the security measures, the security network driver passes the network packets back to the application programming interface.

5. The method of claim 1, wherein the network comprising a public network.

6. The method of claim 1, wherein the security measures include:

encapsulating the network packets before the network packets are transferred to the network.

7. The method of claim 1, wherein the security measures include:

decapsulating the network packets after the network packets are received from the network.

8. The method of claim 1, wherein the security measures include:

encrypting the network packets before the network packets are transferred to the network.

9. The method of claim 1, wherein the security measures include:

decrypting the network packets after the network packets are received from the network.

10. The method of claim 1, wherein the security measures are selectable.

11. The method of claim 10, wherein the security measures are selectable through libraries.

12. The method of claim 11, wherein the libraries include an encapsulation/decapsulation library.

13. The method of claim 11, wherein the libraries include an encryption/decryption library.

14. The method of claim 1, further comprising, before passing network packets between the network protocol and the network driver via the application programming interface:

passing the network packets between an application and the network protocol.

15. The method of claim 1, further comprising, before passing network packets between the network protocol and the network driver via the application programming interface:

passing network packets between an internal network driver and the network protocol.

16. A method for processing network packets communicated on a network, comprising:

passing the network packets from a network protocol to an application programming interface;

passing the network packets from the application programming interface to a security network driver;

performing security measures on the network packets;

passing the network packets from the security network driver back to the application programming interface;

passing the network packets to a network driver; and

sending the network packets over the network.

17. The method of claim 16, further comprising, before passing network packets from the network protocol to the application programming interface:

passing network packets from an application or an internal network driver to the network protocol.

18. The method of claim 16, further comprising:

receiving network packets over the network;

passing the network packets from the network driver to the application programming interface;

passing the network packets from the application programming interface to the security network driver;

performing security measures on the network packets;

passing the network packets from the security network driver back to the application programming interface; and

passing the network packets from the application programming interface to the network protocol.

19. The method of claim 18, further comprising:

passing the network packets from the network protocol to an application or an internal network driver.

20. A method for use with a network protocol application programming interface, comprising:

passing network packets to a security network driver from the application programming interface;

performing security measures on the network packets; and

passing the secure network packets back to the application programming interface.

21. The method of claim 20, further comprising:

passing network packets between a network protocol and the application programming interface.

22. The method of claim 20, further comprising:

passing network packets between a network driver and the application programming interface.

23. The method of claim 20, further comprising, before passing network packets to the security network driver:

altering a roadmap to allow network packets to be passed between the application programming interface and the security network driver.

24. The method of claim 20, wherein the network comprises a public network.

25. A network packet processor, comprising:

a network protocol;

an application programming interface coupled with the network protocol and configured to pass network packets with the network protocol;

a security network driver coupled with the application programming interface and configured to pass the network packets with the application programming interface and perform security measures on the network packets; and

a network driver coupled with the application programming interface and configured to pass the network packets with the application programming interface.

26. The network packet processor of claim 25, further comprising:

an application coupled with the network protocol and configured to pass the network packets with the network protocol.

27. The network packet processor of claim 25, further comprising:

an internal network driver coupled with the network protocol and configured to pass the network packets with the network protocol.

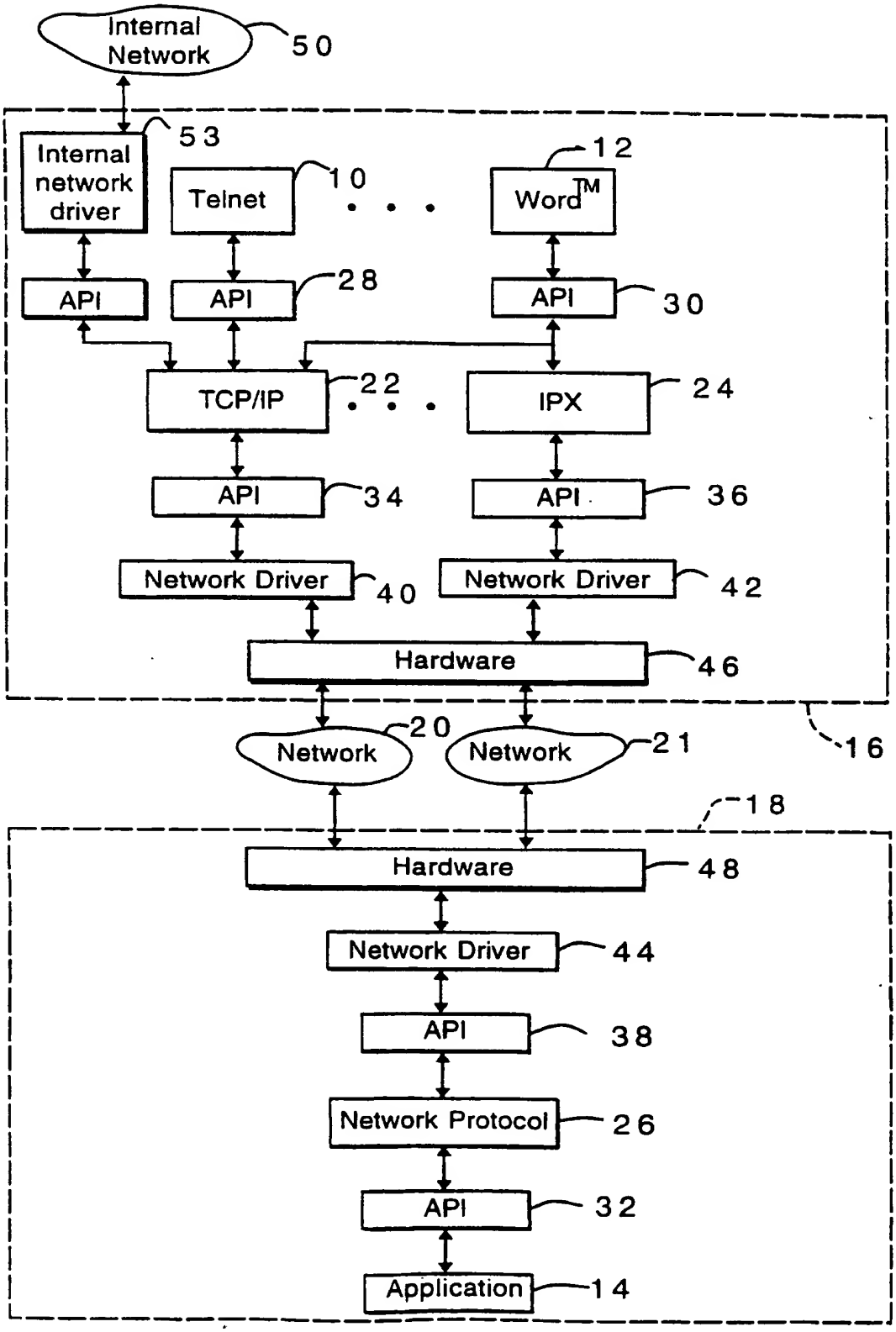

Internal Network
50
Internal network driver
53
Telnet
10
Word™
12
API
API
28
API
30
TCP/IP
22
IPX
24
API
34
API
36
Network Driver
40
Network Driver
42
Hardware
46
16
Network
20
Network
21
18
Hardware
48
Network Driver
44
API
38
Network Protocol
26
API
32
Application
14

FIG. 1

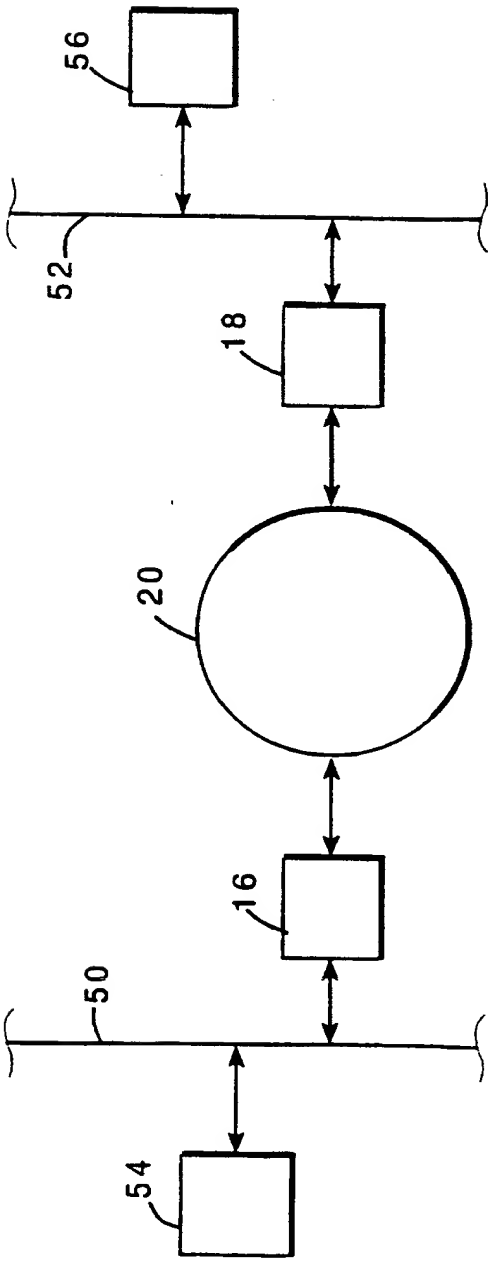


FIG. 2

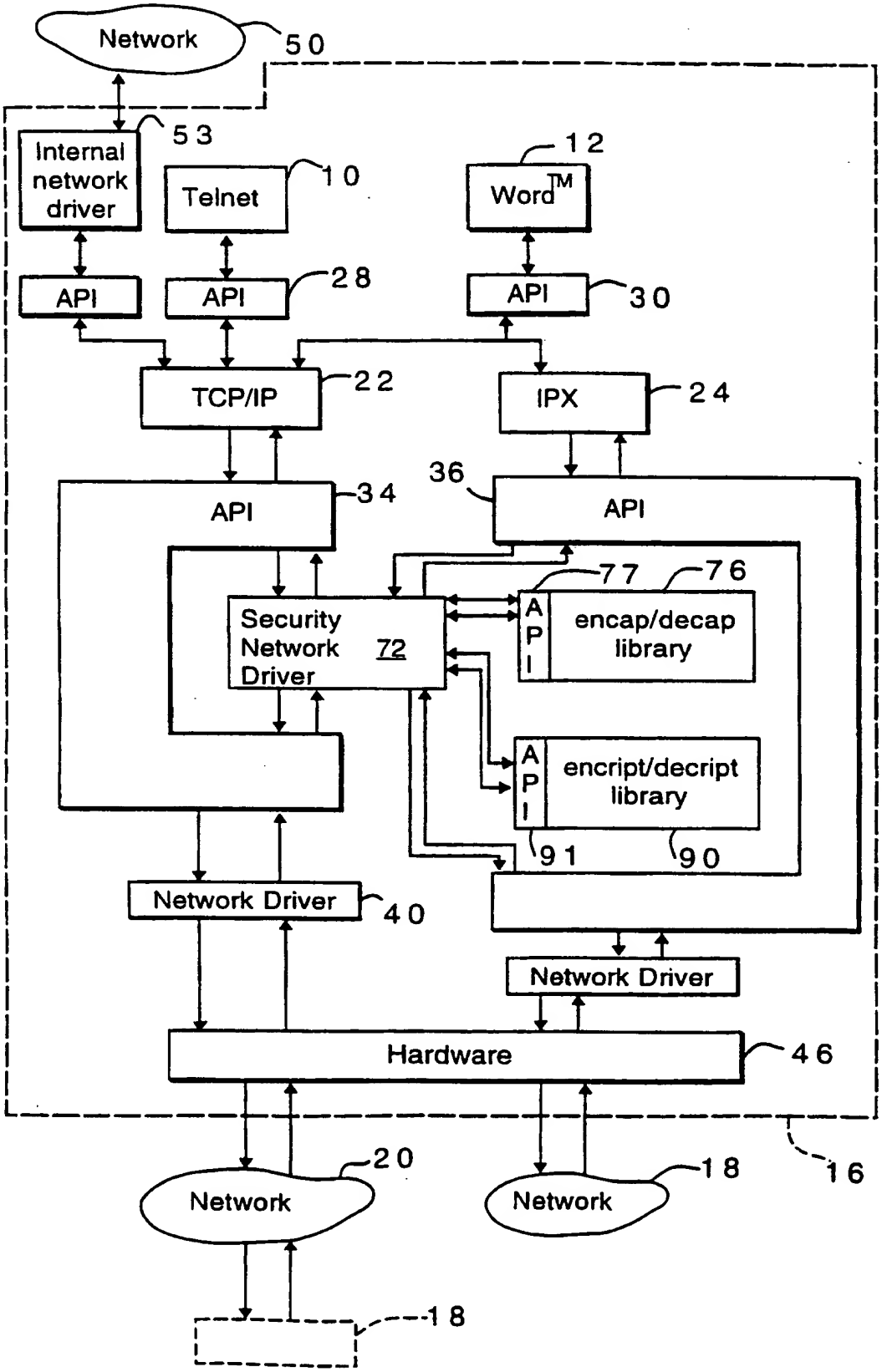

Network
50
Internal network driver
53
Telnet
10
Word™
12
API
API
28
API
30
TCP/IP
22
IPX
24
API
34
36
API
77
76
Security Network Driver 72
API
encap/decap library
API
encript/decript library
91
90
Network Driver
40
Network Driver
Hardware
46
16
Network
20
Network
18
18

FIG. 3

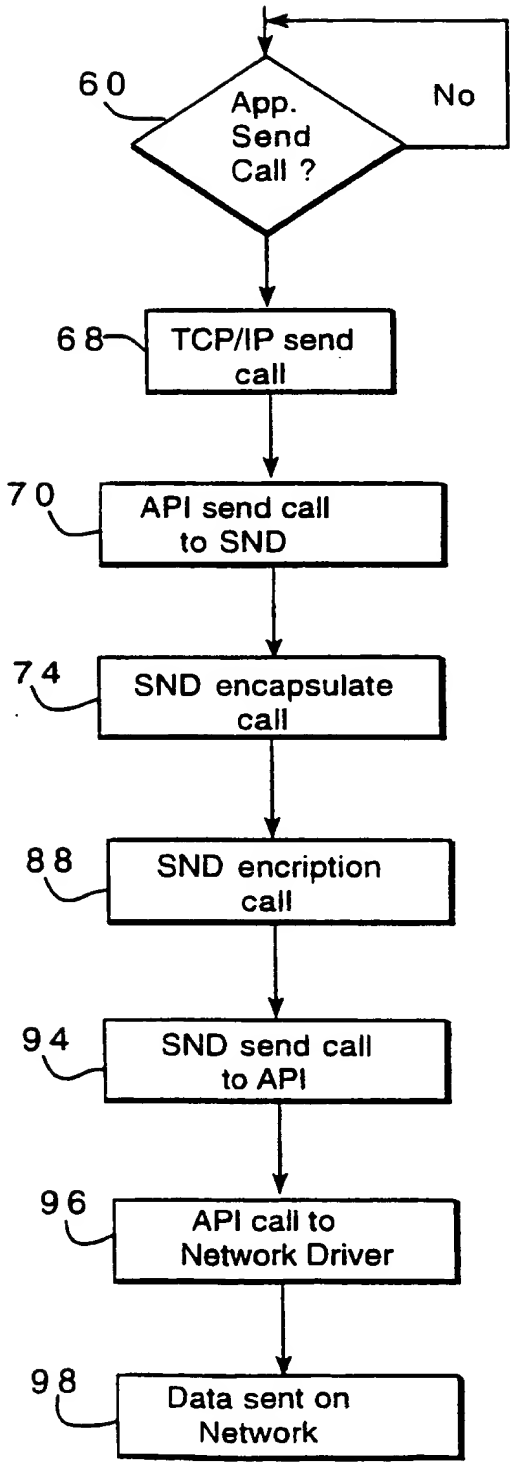

60
App. Send Call ?
No
68
TCP/IP send call
70
API send call to SND
74
SND encapsulate call
88
SND encription call
94
SND send call to API
96
API call to Network Driver
98
Data sent on Network

FIG. 4

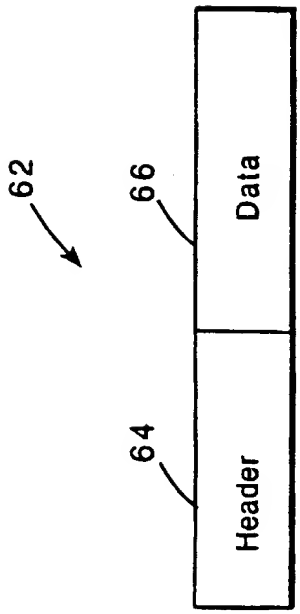

62
66
Data
64
Header

FIG. 5

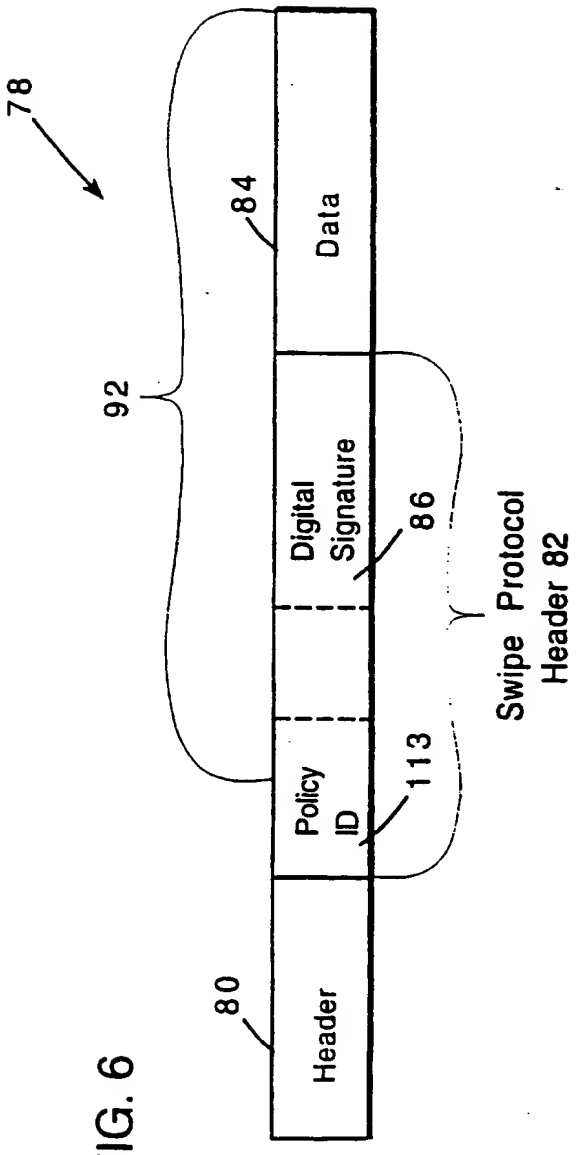

78
84
Data
92
Digital Signature
86
Swipe Protocol Header 82
Policy ID
113
80
Header

FIG. 6

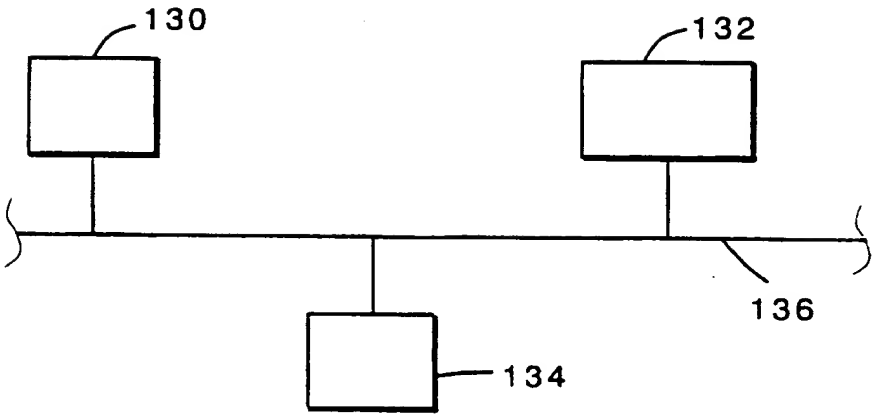


FIG. 9

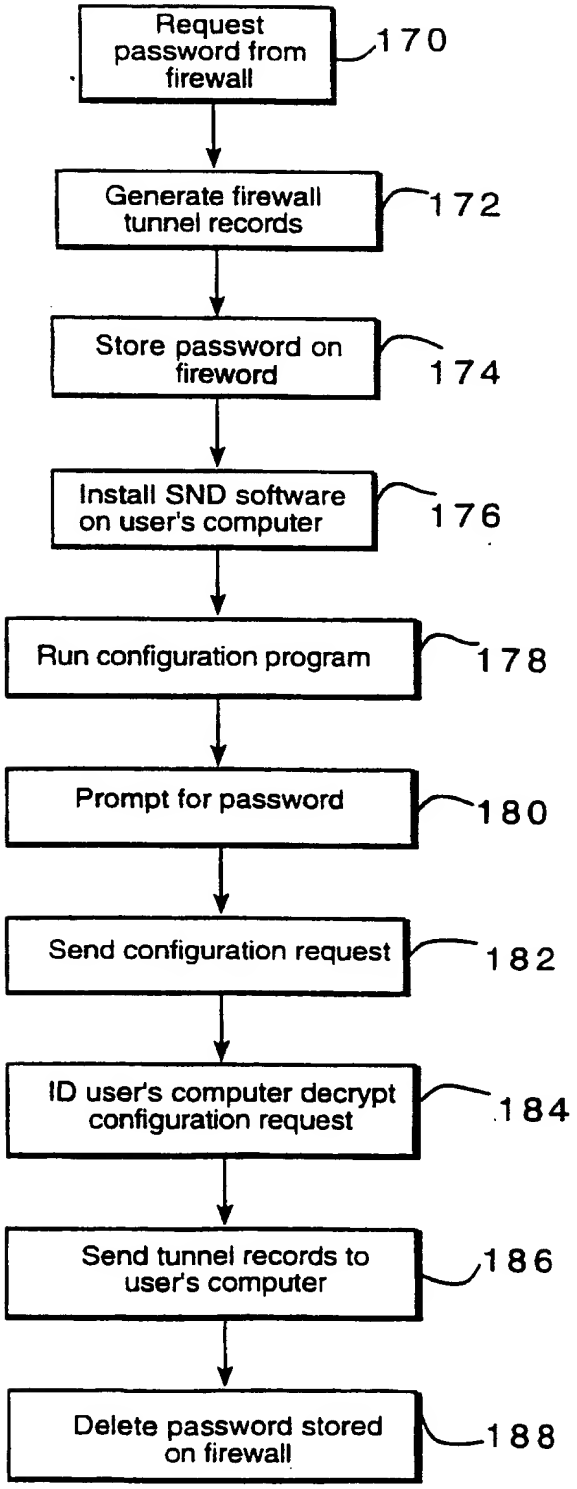

Request password from firewall
170
Generate firewall tunnel records
172
Store password on fireword
174
Install SND software on user's computer
176
Run configuration program
178
Prompt for password
180
Send configuration request
182
ID user's computer decrypt configuration request
184
Send tunnel records to user's computer
186
Delete password stored on firewall
188

FIG.10

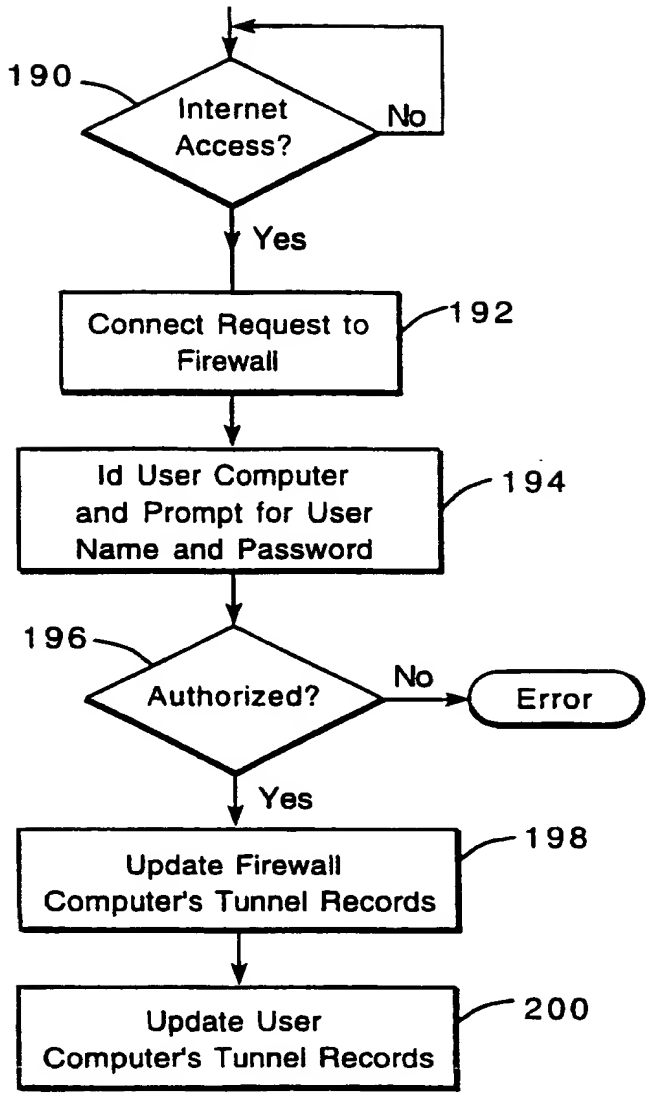

190
Internet Access?
No
Yes
Connect Request to Firewall
192
Id User Computer and Prompt for User Name and Password
194
196
Authorized?
No
Error
Yes
Update Firewall Computer's Tunnel Records
198
Update User Computer's Tunnel Records
200

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/00640

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :H04L 9/00
US CL : 380/49

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 380/49, 9, 23, 25, 48, 50, 59

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,086,469 A (GUPTA et al) 04 February 1992, see Abstract.	1-27
X	US 5,416,842 A (AZIZ) 16 May 1995, see Abstract.	1-27

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 04 APRIL 1997	Date of mailing of the international search report 30 APR 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer BERNARR EARL GREGORY Telephone No. (703) 306-4153

Form PCT/ISA/210 (second sheet)(July 1992)★